REMARKS

In response to the Final Office Action dated October 17, 2006, Applicants respectfully request reconsideration based on the above claim amendments and the following remarks. Applicants respectfully submit that the claims as presented are in condition for allowance. Reconsideration of the present application is respectfully requested in view of the following remarks. Prior to entry of this response, Claims 1-20 were pending in the application, of which Claims 1, 8, and 14 are independent. In the Final Office Action dated October 17, 2006, Claims 1-20 were rejected under 35 U.S.C. § 103(a). Following this response, Claims 1-18 and 20 remain in this application, with Claim 19 being canceled without prejudice or disclaimer. Applicants hereby address the Examiner's rejections in turn.

I. Rejection of the Claims Under 35 U.S.C. § 103(a)

In the Final Office Action dated October 17, 2006, the Examiner rejected Claim 1-18 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,999,948 ("*Nelson*") in view of U.S. Patent No. 6,496,843 ("*Getchius*"). Claims 1, 8, and 14 have been amended, and Applicants respectfully submit that the amendments overcome this rejection and add no new matter.

Amended Claim 1 is patentably distinguishable over the cited art for at least the reason that it recites, for example, "wherein the one or more field names are identical to corresponding fields in the form and the one or more field names are associated with a corresponding response data of the form, wherein a software component is not hard-coded with the one or more field names." Amended Claims 8 and 14 each includes a

similar recitation. Support for the amendment can be found in the specification at least page 2, lines 25-27 and page 3, lines 15-25.

Consistent with an embodiment of the invention, when a class file is compiled, a field engine table may be consulted and the field names of fields to be placed on a requested form may be retrieved. (*See* specification, page 3, lines 9-10.) The filed names specified in the filed engine table may then be associated with corresponding fields in the form. (*See* specification, page 3, lines 10-12.) Markup language for displaying the form may be constructed so the filed names may be returned with corresponding form response data when the completed form is submitted to a software component. (*See* specification, page 3, lines 12-14.) The software component may also receive a submission of response data associated with each off the field names. (*See* specification, page 3, lines 15-16.) The receiving of the submission of response data may occur in response to a submission of the completed form by a user. (*See* specification, page 3, lines 16-17.)

In contrast, *Nelson* at least does not disclose the aforementioned recitation. *Nelson* merely discloses factory objects form specification classes and catalogued forms. For example, in *Nelson*, a user identifies data to be presented in a form. (*See* col. 5, lines 37-38.) The user then writes a FDL file. (*See* col. 5, line 38.) Next, in *Nelson*, an application using dynamic forms registers the FDL file with a dynamic forms engine using a client API. (*See* col. 5, lines 44-45.) The FDL file is parsed to generate necessary factory objects and form specification classes. (*See* col. 5, lines 47-49.) *Nelson's* factory objects and form specification classes are then added to a list of catalogued forms. (*See* col. 5, lines 52-52.) Consequently, parsing an FDL file does

not disclose associating non-hard-coded field names with corresponding response data. Because *Nelson* parses FDL files in order to create factory objects, form specification classes, and catalogued forms, *Nelson* does not disclose non-hard-coded field names that are identical to corresponding fields in the form and associated with corresponding form response data.

Furthermore, *Getchius* does not overcome *Nelson's* deficiencies. *Getchius* merely discloses that the external process may copy blob data from multiple tables in which the associated field name differ with each table. (*See* col. 52, lines 31-33.) The external process uses the data included in a temporary table 1242 to fetch or access the blob data associated with a particular table name and field name to subsequently index into each particular table name using the identifier to extract the actual blob data. (*See* col. 52, lines 38-42.) The blob data is copied to a repository table 1226 on a receiving node. (*See* col. 52, lines 42-44.) In *Getchius*, the data described in various tables are functional equivalents. (*See* col. 52, lines 55-58.) For example, a table 1224 includes a blob pointer field that acts as an index into repository table 1226, whereas a table 1220 includes the actual blob data in a field. (*See* col. 52, lines 58-61.) Therefore, in *Getchius*, using the blob pointer field in table 1224 acts as an index into repository table 1226. Consequently, using a pointer as an index does not disclose associating non-hard-coded identical field names with corresponding response data. In other words, because *Getchius* uses pointers, *Getchius* does not disclose non-hard-coded field names that are identical to corresponding fields in the form and associated with corresponding form response data.

Combining *Nelson* with *Getchius* would not have led to the claimed invention because *Nelson* and *Getchius*, either individually or in any reasonable combination, at least do not disclose "wherein the one or more field names are identical to corresponding fields in the form and the one or more field names are associated with a corresponding response data of the form, wherein a software component is not hard-coded with the one or more field names," as recited by amended Claim 1. Amended Claims 8 and 14 each includes a similar recitation. Accordingly, independent Claims 1, 8, and 14 each patentably distinguishes the present invention over the cited art, and Applicants respectfully request withdrawal of this rejection of Claims 1, 8, and 14.

Dependent Claims 2-7, 9-13, and 15-18 are also allowable at least for the reasons described above regarding independent Claims 1, 8, and 14, and by virtue of their respective dependencies upon independent Claims 1, 8, and 14. Accordingly, Applicants respectfully request withdrawal of this rejection of dependent Claims 2-7, 9-13, and 15-18.

II. Rejection of the Claims Under 35 U.S.C. § 103(a)

In the Final Office Action, the Examiner rejected Claim 19-20 under 35 U.S.C. § 103(a) as being unpatentable over *Nelson* in view of *Getchius* further in view of U.S. Patent No. 6,718515 ("*Conner*"). Claim 19 has been canceled without prejudice or disclaimer.

Dependent Claim 20 is patentably distinguishable over the cited art for at least for the reason that it recites, due to its dependency on amended independent Claim 1, "wherein the one or more field names are identical to corresponding fields in the form

and the one or more field names are associated with a corresponding response data of the form, wherein a software component is not hard-coded with the one or more field names."

As stated above, consistent with an embodiment of the invention, if a previously compiled class file cannot be utilized, a software component compiles a class file capable of generating fields for a requested form. (*See* specification, page 3, lines 7-8.) When the class file is compiled, a field engine table is consulted and field names of the field to be placed on the requested form are retrieved. (*See* specification, page 3, lines 9-10.) The field names specified in the field engine table are then associated with corresponding fields in the form. (*See* specification, page 3, lines 10-12.) In this manner, markup language for displaying the form is constructed so the filed names will be returned with the corresponding form response data when a completed form is submitted to the software component. (*See* specification, page 3, lines 12-14.)

In contrast, *Nelson* at least does not disclose the aforementioned recitation. *Nelson* merely discloses factory objects form specification classes and catalogued forms. For example, in *Nelson*, a user identifies data to be presented in a form. (*See* col. 5, lines 37-38.) The user then writes a FDL file. (*See* col. 5, line 38.) Next, in *Nelson*, an application using dynamic forms registers the FDL file with a dynamic forms engine using a client API. (*See* col. 5, lines 44-45.) The FDL file is parsed to generate necessary factory objects and form specification classes. (*See* col. 5, lines 47-49.) *Nelson's* factory objects and form specification classes are then added to a list of catalogued forms. (*See* col. 5, lines 52-52.) Consequently, parsing an FDL file does not disclose associating non-hard-coded field names with corresponding response data.

Because *Nelson* parses FDL files in order to create factory objects, form specification classes, and catalogued forms, *Nelson* does not disclose non-hard-coded field names that are identical to corresponding fields in the form and associated with corresponding form response data.

In addition, *Getchius* does not overcome *Nelson's* deficiencies. *Getchius* merely discloses that the external process may copy blob data from multiple tables in which the associated field name differ with each table. (*See* col. 52, lines 31-33.) The external process uses the data included in a temporary table 1242 to fetch or access the blob data associated with a particular table name and field name to subsequently index into each particular table name using the identifier to extract the actual blob data. (*See* col. 52, lines 38-42.) The blob data is copied to a repository table 1226 on a receiving node. (*See* col. 52, lines 42-44.) In *Getchius*, the data described in various tables are functional equivalents. (*See* col. 52, lines 55-58.) For example, a table 1224 includes a blob pointer field that acts as an index into repository table 1226, whereas a table 1220 includes the actual blob data in a field. (*See* col. 52, lines 58-61.) Therefore, in *Getchius*, using the blob pointer field in table 1224 acts as an index into repository table 1226. Consequently, using a pointer as an index does not disclose associating non-hard-coded identical field names with corresponding response data. In other words, because *Getchius* uses pointers, *Getchius* does not disclose non-hard-coded field names that are identical to corresponding fields in the form and associated with corresponding form response data.

Furthermore, *Conner* does not overcome *Nelson's* and *Getchius'* deficiencies. *Conner* merely discloses a method for creating a table format object and using the object to generate an HTML table as a dynamic page in response to a client browser. (*See* col. 5, lines 11-14.) The routine begins by creating a table format object called a tableFormatter. (*See* col. 5, lines 14-16.) The object is created during a page authoring process. (*See* col. 5, lines 16-17.) In response to a client request, the request object and data object are passed, in *Conner*, to the tableFormatter that formats the table for use in a page. (*See* col. 5, lines 38-44.) In other words, in response to a client request, a .jsp servlet creates the HTML table. Then the servlet populates the table according to properties set in the tableFormatter that is hard-coded by a page author. *Conner* populates a table according to properties that are hard-coded and not retrieved. *Conner* does not disclose retrieving field names to populate a table because *Conner* discloses populating a table via hard-coding and not retrieving the names to populate the fields. Like *Nelson* and *Getchius*, *Conner* at least does not disclose non-hard-coded field names that are identical to corresponding fields in the form and associated with corresponding form response data.

Combining *Nelson* with *Getchius* and *Conner* would not have led to the claimed invention because *Nelson*, *Getchius*, and *Conner*, either individually or in any reasonable combination, at least does not disclose "wherein the one or more field names are identical to corresponding fields in the form and the one or more field names are associated with a corresponding response data of the form, wherein a software component is not hard-coded with the one or more field names," as included in dependent Claim 20. Accordingly, dependent Claim 20 patentably distinguishes the

present invention over the cited art, and Applicants respectfully request withdrawal of this rejection of dependent Claim 20.

III. Conclusion

In view of the foregoing remarks, Applicants respectfully request the reconsideration and reexamination of this application and the timely allowance of the pending claims. The preceding arguments are based only on the arguments in the Office Action, and therefore do not address patentable aspects of the invention that were not addressed by the Examiner in the Office Action. The claims may include other elements that are not shown, taught, or suggested by the cited art. Accordingly, the preceding argument in favor of patentability is advanced without prejudice to other bases of patentability. Furthermore, the Office Action contains a number of statements reflecting characterizations of the related art and the claims. Regardless of whether any such statement is identified herein, Applicant declines to automatically subscribe to any statement or characterization in the Office Action.

Please grant any extensions of time required to enter this response and charge any additional required fees to our deposit account 13-2725.

Respectfully submitted,
MERCHANT & GOULD P.C.

P.O. Box 2903
Minneapolis, MN 55402-0903
404.954.5066

Date: January 17, 2007

D. Kent Stier
Reg. No. 50,640

DKS:ARL:mdc